



מתרגל ממונה על התרגיל: איל רavid, oi.ravid@campus.technion.ac.il

תאריך ושעה הגשה: 21/12/2025 בשעה 23:59

בזוגות. אין להגיש ביחידים. (אלא באישור מתרגל אחראי של הקורס).

אוףן ההגשה:

הנחיות כלליות:

- שאלות על התרגיל יש לפרסם באתר הפיאצה של הקורס תחת לשונית "wet_1":
 - האתר: <https://piazza.com/technion.ac.il/winter2026/234218>, נא לקרוא את השאלות של סטודנטים אחרים לפני שמספרם שאלת חדשה, למקורה שנשאלה כבר.
 - נא לקרוא את המסמך "נהלי הקורס" באתר הקורס. בנוסף, נא לקרוא בעיון את כל ההנחיות בסוף מסמך זה.
 - בפורום הפיאצה יונאל FAQ ובמידת הצורך יעלוי תיקונים כהודעות געוזות (Pinned Notes). תיקונים אלו מחייבים.
 - התרגיל מורכב משני חלקים: יש ורטוב.
 - לאחר קראת כל הדרישות, מומלץ לתכנן תחילת את מבני הנתונים על נייר. דבר זה יכול להשורר לכם זמן רב.
 - לפני שתת媚ם ניגשים לקודד את פתרונכם, ודאו כי יש לכם פתרון העומד בכל דרישות הסביבות בתרגיל. תרגיל שני מומלץ לחלק למחלקות שונות שאפשר למש (ולבדוק!) בהדרגות.
 - את הפתרון שלכם מומלץ לחלק למחלקות שונות שאפשר למש (ולבדוק!) בהדרגות.
 - המלצות לפתרון התרגיל נמצאות באתר הקורס תחת: "Programming Tips Session".
 - המלצות לתוכנות במסמך זה אין מחייבות, אך מומלץ להיעזר בהן.
 - חומר התרגיל הינו כל החומר שנלמד בהרצאות ובתרגולים עד אך לא כולל עצי דרגות.
 - העתקת תרגיל בית רטובים תיבדק באמצעות תוכנת בדיקות אוטומטית, המזהה דמיון בין כל העבודות הקיימות במערכת, גם כאלו משנים קודמות. לא ניתן לעתר על החלטת התוכנה. התוכנה אינה מבדילה בין מקור להעתיק! אנא הימנעו מהסתכלות בקוד שאתם שילכם.
 - בקשوت להגשה מאוחרת יש להפנות למתרגל האחראי בלבד בכתב:

goldshtein@campus.technion.ac.il



הקדמה:

בעקבות באג במערכת סאף, נמחקו כל הנתונים על הסטודנטים והקורסים בפקולטה. בזכות הידע הנרחב שצברתם בקורס, מזכירות הפקולטה מבקשת מכם, הסטודנטים לבנות מערכת חדשה בשם TechSystem לניהול מאגר הסטודנטים והקורסים. לזכירות הפקולטה דרישות מאוד גבוהות והמערכת שתבנו תצריך לעמוד בדרישות סיבוכיות ווקשות.



TechSystem

סימוניים לצורכי סיבוכיות:

נסמן ב-*a* את מספר הסטודנט במערכת.

ב-*b* את מספר הקורסים במערכת.

ב-*coursesId* את מספר הסטודנטים הרשומים לקורס בעל המזהה courseId אם אין קורס כזה אפשר להתייחס לערך זה כ-1.

דרוש מבנה נתונים למימוש הפעולות הבאות:

`techSystem()`

מאתחלת מבנה נתונים ריק. תחילת אין במערכת סטודנטים או קורסים.

פרמטרים: אין

ערך החזרה: אין

סיבוכיות זמן: (1) 0 במקרה הגורע.

`virtual ~techSystem()`

הפעולה משחררת את המבנה (כל הזיכרון אותו הקצתתם חייב להיות משוחרר).

פרמטרים: אין

ערך החזרה: אין

סיבוכיות זמן: (m · n) 0 במקרה הגורע.

מבני נתונים 23421 חורף תשפ"ו

gilion rotov מס' 1 – מעודכן לtarיך 24.11.2025

עמוד 3 מתוך 8



StatusType addStudent(int studentId)

הפעולה מוסיפה לבנייה נתונים סטודנט בעל מזהה *studentId* עם 0 נק"ג. הסטודנט לא רשום לאף קורס.

פרמטרים:

studentId ערך החזרה:

במקרה של בעיה בהקצאה/שחרור זיכרון. ALLOCATION_ERROR

.studentId אם $<=0$ INVALID_INPUT

.studentId אם קיימן כבר סטודנט עם זהה Id FAILURE

.studentId במקרה של הצלחה. SUCCESS

סיבוכיות זמן: $O(\log n)$ במקרה הגרוע.

StatusType removeStudent(int studentId)

הסטודנט בעל זהה *studentId* סימן את התואר ו开会 ציר להוציאו מהמערכת. אם הסטודנט רשום לפחות אחד או יותר קורסים אז לא נוציאו אותו מהמערכת והוא עוד לא יסימן את התואר.

פרמטרים:

studentId ערך החזרה:

במקרה של בעיה בהקצאה/שחרור זיכרון. ALLOCATION_ERROR

.studentId אם $<=0$ INVALID_INPUT

.studentId אם אין סטודנט עם זהה Id או שהסטודנט רשום לפחות קורסים. FAILURE

.studentId במקרה של הצלחה. SUCCESS

סיבוכיות זמן: $O(\log n)$ במקרה הגרוע.

StatusType addCourse(int courseId, int points)

קורס בעל זהה ייחודי *courseId* מתווסף למערכת, סיום הקורס מזכה את הסטודנט המסיים ב נק"ג. תחילת לימודי לא רשומים סטודנטים.

פרמטרים:

courseId ערך החזרה:

זהה הקורס שחייב להוסיף. points

מספר הנק"ג שהקורס מזכה סטודנט שעבר אותו. FAILURE

.points אם $<=0$ INVALID_INPUT

.courseId אם קיימן כבר קורס בזהה courseId במערכת. SUCCESS

.courseId במקרה של הצלחה. SUCCESS

סיבוכיות זמן: $O(m \log m)$ במקרה הגרוע.

StatusType removeCourse(int courseId)

עקב חוסר ביקוש, מבטלים את הקורס בעל זהה courseId ומוציאים אותו מהמערכת. אם יש סטודנטים הרשומים לקורס, אז הקורס לא יצא מהמערכת ולא יבוטל.

פרמטרים:

courseId ערך החזרה:

זהה הקורס שմבקשים להסיר. courseId

במקרה של בעיה בהקצאה/שחרור זיכרון. ALLOCATION_ERROR

.courseId אם $<=0$ INVALID_INPUT

מבני נתונים 23421 חורף תשפ"ו

gilion rotov מס' 1 – מעודכן לtarיך 24.11.2025

עמוד 4 מתוך 8



אם אין קורס עם מזהה <i>courseId</i> או שיש סטודנטים הרשומים לקורס בעל המזהה <i>courseId</i> במקורה של הצלחה.	FAILURE
	SUCCESS

סיבוכיות זמן: $O(\log m)$ במקרה הגרוע.

StatusType *enrollStudent(int studentId, int courseId)*

סטודנט בעל המזהה *studentId* נרשם לקורס בעל המזהה *courseId*.

פרמטרים:

מזהה הסטודנט שרצה להרשם.	<i>studentId</i>
מזהה הקורס אליו הסטודנט נרשם.	<i>courseId</i>

ערך החזרה:

במקרה של בעיה בהקצאה/שחרור זיכרון.	ALLOCATION_ERROR
אם $studentId <= 0$ או $courseId <= 0$	INVALID_INPUT
אם אין סטודנט עם מזהה <i>studentId</i> או אין קורס עם מזהה <i>courseId</i> במערכת או שהסטודנט כבר רשום לקורס אליו הוא מנסה להרשם.	FAILURE
במקרה של הצלחה.	SUCCESS

סיבוכיות זמן: $O(n \log m + \log m)$ במקרה הגרוע.

StatusType *completeCourse(int studentId, int courseId)*

סטודנט בעל המזהה *studentId* השלים את חובותיו בקורס בעל המזהה *courseId* ולכון הוא מקבל את הנק"ז של הקורס ומוצא מהקורס.

פרמטרים:

מזהה הסטודנט שהשלים את הקורס.	<i>studentId</i>
מזהה הקורס שאוינו הסטודנט השלם.	<i>courseId</i>

ערך החזרה:

במקרה של בעיה בהקצאה/שחרור זיכרון.	ALLOCATION_ERROR
אם $studentId <= 0$ או $courseId <= 0$	INVALID_INPUT
אם אין קורס בmaze <i>courseId</i> או אין סטודנט בעל המזהה <i>studentId</i> או <i>studentId</i> לא רשום לקורס עם המזהה <i>courseId</i> .	FAILURE
במקרה של הצלחה.	SUCCESS

סיבוכיות זמן: $O(\log m + \log n_{courses})$ במקרה הגרוע.

StatusType *awardAcademicPoints(int points)*

דיקן הפקולטה החליט לתת בונוס לכל הסטודנטים בפקולטה של *points* נק"ז במתנה. הבונוס תקין אך ורק לסטודנטים שנמצאים בזמן הקרייה לפונקציה ולא משפיע על הנק"ז של סטודנטים שיצטרפו בעתיד.

פרמטרים:

מספר הנק"ז שהדיקן מעניק לסטודנטים.	<i>points</i>
------------------------------------	---------------

ערך החזרה:

במקרה של בעיה בהקצאה/שחרור זיכרון.	ALLOCATION_ERROR
אם $points <= 0$	INVALID_INPUT
במקרה של הצלחה.	SUCCESS

סיבוכיות זמן: $O(1)$ במקרה הגרוע.



```
output_t < int > getStudentPoints(int studentId)
```

מחזיר את מספר הנק"ז שצבר הסטודנט בעל מזהה *studentId*

פרמטרים:

מזהה הסטודנט שצሪ להחזיר את מספר הנק"ז שצבר.

studentId

ערך החזרה:

במקרה של בעיה בהקצאה/שחרור זיכרון.

אם *studentId* ≤ 0 INVALID_INPUT

אם אין סטודנט בעל מזהה *studentId* FAILURE

במקרה של הצלחה, במקרה זה יוחזר גם מספר הנק"ז של הסטודנט.

SUCCESS

סיבוכיות זמן: $O(\log n)$ במקרה האגוף.

**דוגמיה הרצה:**

```

addCourse(1, 3): SUCCESS
addCourse(2, 5): SUCCESS
addStudent(1): SUCCESS
addStudent(5): SUCCESS
getStudentPoints(1): SUCCESS, Value: 0
enrollStudent(1, 1): SUCCESS
removeStudent(1): FAILURE
removeCourse(1): FAILURE
getStudentPoints(1): SUCCESS, Value: 0
completeCourse(1, 1): SUCCESS
completeCourse(1, 2): FAILURE
removeCourse(1): SUCCESS
getStudentPoints(1): SUCCESS, Value: 3
awardAcademicPoints(10): SUCCESS
getStudentPoints(1): SUCCESS, Value: 13
removeStudent(1): SUCCESS
addStudent(100): SUCCESS
getStudentPoints(100): SUCCESS, Value: 0

```

סיבוכיות מקום:

סיבוכיות המקום הדרישה עבור מבנה הנתונים היא ($n_i + m + \sum_{i \text{ is a course}} n_i$) 0 במקורה הגרווע. כלומר בכל רגע בזמן הריצה, ציריכת המקום של מבנה הנתונים תהיה לנארית בסכום מספרי הסטודנטים, הקורסים במערכת ומספר הסטודנטים בכל הקורסים יחד כולל כפליות.

סיבוכיות המקום הנדרשת עבור כל פעולה (לצורך, זיכרנו "העזר" שככל פעולה משתמש בו) אינה מצוינת לכל פעולה לחוד, אך אסור לעבור את סיבוכיות המקום הדרישה שמודדרת לכל המבנה.

ערכי החזרה של הפונקציות:

כל אחת מהfonקציות מחזירה ערך מטיפוס StatusType שייקבע לפי הכלל הבא:

- תחיליה, יוחזר INVALID_INPUT אם הקלט אינו תקין.
 - אם לא הוחזר INVALID_INPUT:
 - בכל שלב בפונקציה, אם קורתה שגיאת הקצאה/שחרור יש להחזיר ALLOCATION_ERROR. מצב זה אינו צפוי אלא באחד משני מקרים (לרוב): באמצעות השימוש בזיכרון גדול מאוד ולן המבנה ניצל את כל ההזיכרונות במערכת, או שיש זליגת זיכרון בזיכרון.
 - אם קורתה שגיאיה אחרת, כפי שמצוין בכל פונקציה, יש להחזיר מיד FAILURE מבל' לשנות את מבנה הנתונים.
 - אחרת, יוחזר SUCCESS.
- חלק מהfonקציות ציריכות להחזיר לבסוף עוד פרמטר (int או bool), لكن הן מחזירות אובייקט מטיפוס <T>_output_. אובייקט זה מכיל שני שדות: הסטטוס (status) ושדה נויס (ans) מסוג T.



במקרה של הצלחה (SUCCESS), השדה הנוסף יכול את ערך החזרה, והסטטוס יכול את SUCCESS. בכל מקרה אחר, הסטטוס יכול את סוג השגיאה והשדה הנוסף לא מעניין.

שני הטיפוסים (`output_t<T>, StatusType`) מוממשים כבר בקובץ "util.h" שניתן לכם כחלק מהתרגיל.

קיים קונסטרקטור של `<T> output_t<T>` מ-`T` ומ-`Type`. `output_t` פשוט כתוב בפונקציות הרלוונטיות:
`return 7;`

```
return StatusType::FAILURE;
```

הנחיות וdagשים כלליים:

חלק ייש:

- החלק היבש מהוות חלק מהצון על התרגיל כפי שמצוין בנהלי הקורס.
- לפני שימוש הפעולות בקוד יש לתקן היבט את מבני הנתונים והאלגוריתמים ולעודד כי אפשרותם למש את הפעולות בדרישות הזמן והזיכרון שלו.
- החלק היבש חייב להיות מוקולד.
- הגשת החלק הרטוב מהוות תנאי הכרחי לקבלת ציון על החלק היבש, כמובן, הגשה בה יתקבל איך ורק חלק ישב תגרור ציון 0 על התרגיל כולו.
- שלחcin מסמן הכלול תואר של מבני הנתונים והאלגוריתמים בהם השתמשת ביצוף הוכחות סיבוכיות הזמן והמקום שלהם. חלק זה עומדת בפני עצמה וצריך להיות מובן לקורא גם לפני העיון בקוד. אין צורך לתאר את הקוד בرمת המשתנים, הפונקציות והמחלקות, אלא ברמה העקרונית. חלק ישב זה לא תיעוד קוד.
- הראשית הציגו את מבני הנתונים בהם השתמשתם. רצוי ומומלץ להיעזר בצייר.
- לאחר מכן סבירו כיצד ימשתם כל אחת מהפעולות גורמות לבני הנתונים.
- פועלה תוך כדי התיחסות לשינויים שהפעלות גורמות לבני הנתונים.
- הוכיחו שמבנה הנתונים וכל הפעולות עומדים בדרישת סיבוכיות המקום.
- החסמים הנתונים בתרגיל הם לא בהכרח הדוקים ולכן יכול להיות שקיים פתרון בסיבות טוביה יותר. מספיק להוכיח את החסמים הדרושים בתרגיל.
- רמת פירוט: יש להסביר את כל הפרטים שאינם טריוויאליים ושהסבירים לצורך שימוש הפעולות ועמידה בדרישות הסיבוכיות. אין לדון בפרטים טריוויאליים (הפעילו את שיקול דעתכם בקשר לזה, ושאלו את האחראי על התרגיל אם איןכם בטוחים). אין לצעט קטעים מהקוד כתחליף להסביר. אין צורך לפרט אלגוריתמים שנלמדו בכתה. כמו כן, אין צורך להוכיח תוצאות ידועות שנלמדו בכתה, אלא מספיק לציין בבירור לאיזו תוצאה אתם מתחכו. אין (וגם אין צורך) להשתמש בתוצאות של עצי דרגות ולהלאה.
- **על חלק זה לא לצאת מ-8 עמודים.**
- והכי חשוב **keep it simple!**

חלק רטוב:

- מומלץ למשתתphen תחיליה את מבני הנתונים בצורה הכללית ביותר ורק אז למש את הפונקציות הנדרשות בתרגיל.
- אנו ממליצים בחום על שימוש **Object Oriented**, ב-**++C**, מימוש זה יאפשר לכם להציג לפתרון פשוט וקצר יותר לפונקציות אותן יכולים למש ויאפשר לכם להקל בקבילות את מבני הנתונים שלכם (זכרו שיש תרגיל רטוב נוספת בהמשך הסמסטר).
- פקודות הkompile שモוצאת בעמודת greadescoper הינה: `g++ -std=c++14 -DNDEBUG -Wall -o main.out main.cpp`.
- חתימות הפונקציות שעיכם למש ומספר הגדרות נמצאים בקובץ TechSystem26a1.h.
- אין לשנות את הקבצים `main26a1.cpp` ו-`main26a1.h` אשר סופקו בקורס חלק מהתרגיל, ואין להציג אותן.
- בדיקה אוטומטית שאין בקוד שימוש ב-`STL`, ובדיקה זו נולפת אם מגדים גם את `main26a1.cpp`.
- את שאר הקבצים ניתן לשנות, ותוכלו להוסיף קבצים נוספים כרצונכם, ולהציג אותן.
- העיקר הוא שהקוד שאתם מגישים יתكمפל עם הפקודה לעיל, כאשר מוסיפים לו את שני הקבצים `wet1util.h` ו-`main25b1.cpp`.
- עליים למש בעצמכם את כל מבני הנתונים (למשל **לא** למשתתphen מבנים של **STL** ואין להוריד מבני נתונים מהאנטרכט). **חלק מהתהילה הבדיקה** אנו נבצע בדיקה ידנית של הקוד ונודע שאכן מימושם את מבני הנתונים שבhem השתמשתם.
- בפרט, אסור למשתתphen ב-`zIterator`, `std::pair`, `std::vector`, `std::pair::pair`, או כל אלגוריתם של `STL`, רשימה מלאה של הספריות להן אסור לעשות include נמצאת בקובץ `include.txt`.
- ניתן למשתתphen במצביים חכמים (shared_ptr, Smart pointers) בספרית math או בספרית exception.



חשוב לוודא שאתם מזמנים/משחררים זיכרון בצורה נכונה (ומולץ לוודא עם valgrind). לא חייבים לעבוד עם מצביעים חכמים, אך אם אתם מחליטים כן לעשות זאת, לוודא שאתם משתמשים בהם נכון. (תזכרו שהם לא פתרו קסם, למשל, כאשר יוצרים מנגנון בהצבעות). שימו לב שהעתקת מצביעים חכמים היא איטית מאד וכן יכולה לגרום ל-timeout. עדיף להعبر `tkinter.shared` נ謝רים reference-by.

- **שגיאות של ALLOCATION_ERROR** בד"כ מעידות על דילוג בזיכרון.
- על הקוד להתקמפל ולעboro את כל הבדיקות ש郿ורסות לכם ב-`gradescope`. הטסטים שמורצים באתר מייצגים את הבדיקה אותן נרץ בניתנת הציווין, כאשר פרסמננו 5 מתוך 50.
- אוטם טסטים שבפערם גם מפזריםים קבצי קולט ופלט, יחד עם סקריפט בשם `py_chun` שנכתב בשבייל `python 3.6` ומעלה, המאפשר לבדוק את הקוד שלכם. מומלץ לבדוק את התרגיל לקובאלית לפניה שmagisim. במידה יש `timeout` על אחד מהטסטים זה ייחס ככשלון בטעות. עליים לכתוב קודיעיל במידת הסבר. אין לדאוג לכך יותר מדי, הרוב המוחלט של הפתרונות שעמד בבעיות ריבים ושונים מקבצי הדוגמא הנ"ל. יחד עם זאת הטסטים האלה מייצגים מבחינת אורך ואופן הייצור שלהם את השאר.

אוף הגשה:

הגשת התרגיל מתבצעת דרך אתר ה- `gradescope` של הקורס.

חלק הרטוב:

- יש להגיש רק את קבצי הקוד שלכם (לרוב קבצי `cpp`, `h`, בלבד אפשר גם להגיש אותם כקובץ `dz`) ב- `gradescope` תחת המשימה "wet 1 code part".

חלק היבש:

יש להגיש קובץ PDF אשר מכיל את הפתרון היבש ב- `gradescope` תחת המשימה "wet 1 dry".

שימוש לב שהחלק היבש חייב להיות מוקדם.

- **שימוש לב כי אתם מגישים את כל شيء החלקים הנ"ל, במTELות השונות.**
- לאחר שהגשתם, יש לאפשרותכם לשנות את התוכנית ולהגיש שוב. הגשה האחורונה היא הנחשbeta.
- הערצת הציווין שמצויה ב-`gradescope` אינה ציינכם הסופי על המטלה. הציווין הסופי יתפרסם רק לאחר ההגשתה המאוchorות של משרות המילאים.
- במידה ואתם חושבים לשינה תקלה מהותית במערכת הבדיקה ב-`gradescope` ונא להעלות זאת בפורום הפיאצה ונטפל בה בהקדם.

דחיות ואיוחורים בהגשה:

- דחיות בתרגיל הבית תינטנה אך ורק לפני תקנון הקורס.
- 5 נקודות יורדו על כל יום איוחור בהגשה ללא אישור מראש. לאפשרותכם להגיש תרגיל באיחור של עד 5 ימים ללא אישור. תרגיל שיוגש באיחור של יותר מ-5 ימים ללא אישור מראש יקבל 0.
- במקרה של איוחור בהגשת התרגיל יש עדין להגיש את התרגיל אלקטטרונית דרך האתר הקורס.
- בנסיבות להגשתה מאוחרת יש להפנות למתרגל האחראי בלבד בכתובת `goldshtein@campus.technion.ac.il`. לאחר קבלת אישור במיל על הבקשה, מספר הימים שאושרו لكم נשמר אצלנו. לכן, אין צורך לצרף להגשת התרגיל אישורים נוספים או את שער ההגשה באיחור.

בצלחה!